p. 1, 3, 5-32 (30)

XP-001189063

Jaluna-2 Preview Release 1 Description

Jaluna

Jaluna SA
6, avenue Gustave Eiffel
78180 Montigny-Le-Bretonneux
France

JL/TR-02-92.1
December 2002

Table of Contents

List of Tables

List of Figures

Preface

1. How This Document is Organized

The *Jaluna-2 Preview Release 1 Description* is organized as follows:

- Chapter 1, *Overview*, provides a general introduction to Jaluna-2 PR1.
- Chapter 2, *Cross-Development Environment*, describes the architecture of cross-compilers that forms the Jaluna-2 PR1 cross-development environment.
- Chapter 3, *C5 Microkernel Component*, describes the C5 realtime microkernel component of Jaluna-2 PR1.
- Chapter 4, *Linux Kernel*, describes the Linux kernel component of Jaluna-2 PR1.
- Chapter 5, *Linux Root File System*, describes the Linux root file system and its adaptation to the needs in terms of size and functionality.

2. Related Documents

This book provides a global overview of Jaluna-2 PR1. For precise information regarding installation, development, implementation, deployment and administration, please consult the relevant documents of the rest of the Jaluna-2 PR1 documentation collection.

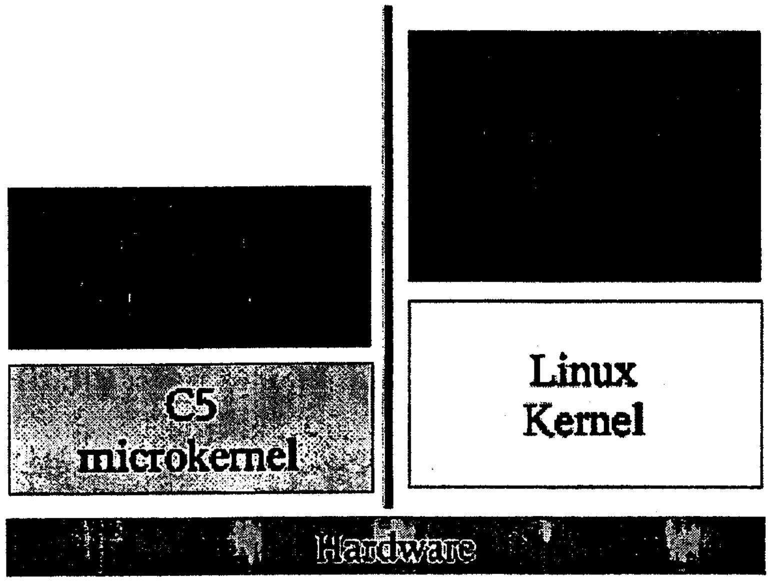
Chapter 1. Overview

1.1. Services and Properties

1.1.1. Dual Operating System

Jaluna-2 Preview Release 1 is an innovative operating system which adds real-time and high availability capabilities to Linux by associating it with the C5 real-time microkernel. Linux and C5 run in parallel on the same processor. Linux and its application, on one side, and C5 and its applications, on the other side, are isolated from each other. Critical applications and Linux applications may exchange data over a specific communication mechanism.

Figure 1-1. Jaluna-2 PR1 Architecture



It is possible to configure out Linux from Jaluna-2 PR1 so that it only supports critical applications. One may also configure out C5 and tailor Jaluna-2 PR1 to be a standard Linux system. The communication mechanism is available over a cPCI bus between two Jaluna-2 PR1 nodes, whatever their configuration is.

The Linux side of Jaluna-2 PR1 is a standard Linux system which can be extended, patched, or enhanced as usual. Jaluna-2 PR1 is a Linux distribution, that is, it also includes a set of system software to make up a full Linux system. Off-the-network additional packages can be built and used as with other distributions.

The C5 side of Jaluna-2 PR1 offers a real-time IP stack configurable through Linux standard interfaces, offering two separate paths for critical and non-critical data.

The isolation between the two sides, and the communication mechanism between them, are the basis for unique Linux high availability features. These features allow Linux applications to efficiently recover from an otherwise fatal Linux failure.

1.1.2. Integrated Development Environment

The Jaluna-2 PR1 cross-development environment provides host-target development tools, leveraging the C5 production environment, enhanced for Linux. It supports C and C++, and covers the software cycle:

- Build of the Jaluna-2 PR1 system: the C5 microkernel, the Linux kernel and the root file system
- Development of drivers, modules and applications whether for C5 or Linux
- Configuration of the system and of the root file system
- Debugging of the system and of the applications

It is therefore possible to generate a full Jaluna-2 PR1 environment from a host system for a target system of a different architecture, without requiring a target system to be connected. The host is a Linux/Intel station, and the target may be for example a PowerPC based system.

On the other hand, when the target is powerful enough and configured properly with appropriate packages and software, it is possible to get part of the compilation environment available on the target itself. This is especially useful when the build of an off-the-network software package cannot cope with cross-compilation.

1.1.3. Configurability

Jaluna-2 PR1 is configurable in the following ways:

System Image

The system image packages in one file the binaries needed to boot up Jaluna-2 PR1. It is composed of memory banks which contain the operating system kernels and possibly other read-only data. Jaluna-2 PR1 uses a flexible, component-based architecture to configure the system image.

C5

The microkernel has a modular architecture. Only the Core Executive component is always present. All features are implemented as components that can be added to, or removed from, an instance of the C5 microkernel.

Linux kernel

As usual a large number of features are available using **make xconfig**.

Linux root file system

The packages to install in the root file system are configured using a dedicated utility.

1.2. Jaluna-2 PR1 Components

1.2.1. C5 Microkernel

The C5 microkernel is almost identical to the one in Jaluna-1 in terms of implementation, functionality and APIs. The only differences are in the low-level layers to allow running aside Linux.

The main features of the microkernel are thread scheduling, multiple synchronization primitives, memory management in several models, time management, actor management. Actors are units of resource allocation, comparable to UNIX processes. All computations outside the kernel (application, device drivers, etc) are performed by actors. The microkernel also implements an advanced device driver toolkit, a fast IPC mechanism, and high availability features. Both the kernel and actors can be debugged at source level (remote gdb) or using an embedded debugger (called kdb).

1.2.2. Linux Kernel

The Linux kernel provided by Jaluna-2 PR1 is based on the reference Linux kernel 2.4.18 from kernel.org (http://www.kernel.org). It comes with additional subdirectories and files containing the appropriate adaptations for running aside C5. A specific kernel module also supports communications with the C5 side. Finally, the x86 kgdb patch is also integrated to support source-level debugging of the Linux kernel.

As a consequence, all drivers, protocols, and services of Linux are kept unchanged. This provides complete applicative compatibility. Applications that rely on precise timeliness are naturally impacted by critical applications running simultaneously on the microkernel, because the tasks of the microkernel are priority tasks.

Configurability of the Linux kernel and standard features are available without modifications in the same state as in the reference kernel: either stable or experimental.

1.2.3. Hardware Resource Allocation

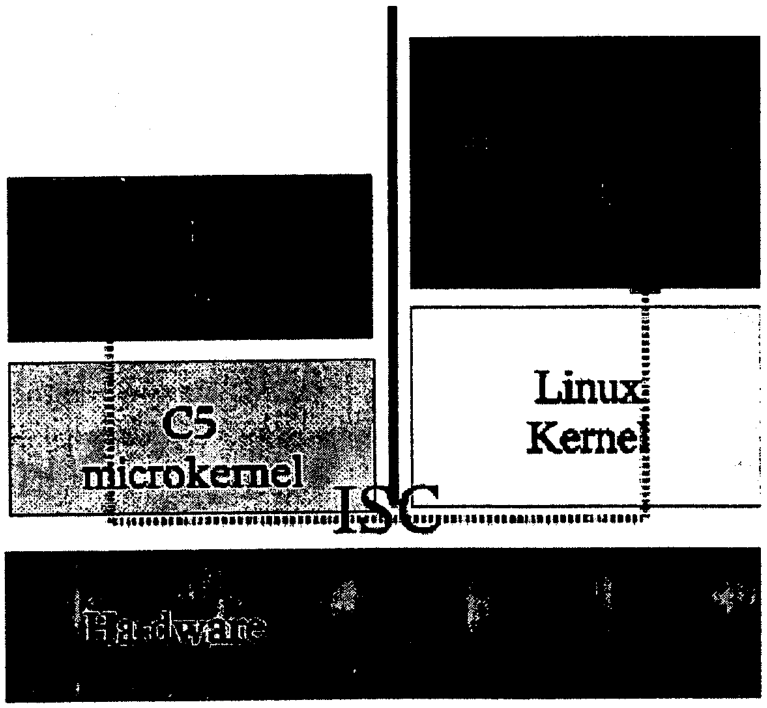
When running concurrently on the same machine, C5 and Linux share resources as follows:

- The CPU is allocated in priority to C5. When C5 is idle, CPU time is allocated to Linux.
- A portion of physical memory is statically allocated to each kernel. This is configurable.
- Other resources are allocated to either one of the OSes: I/O spaces, devices, interrupts. This is configurable.

1.2.4. Inter-System Communication

The Inter-System Communication (ISC) mechanism is dedicated to C5-Linux communications. It implements message-based, reliable, one-way communication channels, available both to applications and from inside the kernels.

Figure 1-2. Inter-System Communication



This communication mechanism is reliable: messages posted are guaranteed to be delivered and will be delivered in order. The maximum size of the messages and their number is defined when establishing a communication channel. Different communication channels may be opened with different maximum message sizes and numbers.

This communication mechanism is designed so that it works in the same reliable fashion whether the two kernels are collocated on the same physical node/board, or whether they run on two distinct boards plugged on the same PCI or cPCI bus.

A Linux kernel module has been developed to provide this service either to other Linux kernel modules, or to Linux applications. C5 also provides an API for accessing this service. Finally, an Ethernet emulation layer over it gives access to the ISC through TCP/IP.

1.2.5. Real-Time UDP/IP Stack

Dedicated to critical applications, the Real-Time UDP/IP stack addresses the following goals:

Preemptability

Packet sending and receiving can be preempted by higher priority tasks.

Determinism

The time spent in the stack, i.e. between the application and the NIC, is bound.

Footprint

The implementation of the stack is memory efficient.

Bandwidth management

Packet writing is partitioned in time slots, to reserve given bandwidth to specific applications.

Small packets

The stack is especially optimized for them.

This stack does not address the problem of hard real-time communications, where the transmission delay itself is predictable.

It is possible to configure this stack from the Linux side. In particular, one may configure this stack so that data directed to some UDP ports be processed on the C5 microkernel side, while the rest of the traffic is forwarded to the Linux full IP stack over the ISC. This allows to share, between C5 and Linux, the same network interface and the same IP address.

1.2.6. High Availability Components

Jaluna-2 PR1 provides two unique high availability components.

The Linux Monitor enables the Linux system to be stopped and restarted, either voluntarily or upon crash, without disturbing the critical applications. The Linux Monitor is composed of a C5 actor and a Linux kernel module. The C5 actor regularly sets a timer, and the Linux kernel module must pat it before the timer expires. When a pat does not occur, Linux is shut down and rebooted. Failures and reboots of the Linux kernel do not impact applications running on C5. Moreover, restart of a Linux system is faster than if Linux was running in a native way on the hardware, since it does not require a hardware reset and Linux is loaded from memory. Time to recover from a failure is therefore reduced.

The Remote Block Driver (RBD) provides Linux applications with persistent memory storage that survives Linux system crashes and reboots. It is composed of a Linux kernel module, and a server running within the C5 microkernel space, talking together. RBD exhibits itself as a disk driver, which may be used as a raw disk or with a file system layered on top of it. The storage is provided by memory allocated on the C5 side. This enables Linux applications to save some state without disk overhead, and to recover after restarting.

1.2.7. Linux Root File System

The Linux root file system is a software set that complements the Linux kernel to provide a full system. This software set comes in source form, packaged as RPM files. The packages provided have been selected to meet embedded system developer requirements. Almost all packages are based on the source RPM packages from the Red Hat 7.3 distribution.

The root file system can be stored on the target's local disk, or mounted from NFS.

In addition to executable files, the packages contain configuration files and initialization scripts run during the initialization of Linux. These scripts have been especially tuned for embedded systems, avoiding reliance on tools typically found on workstation-class machines.

1.3. Application Programming Interfaces

Jaluna-2 PR1 provides multiple sets of programming interfaces, each adapted to a specific usage.

Critical Components:

- Board Support Package Programming Interfaces BKI, DDI, DKI. These APIs are fully compatible with Jaluna-1.
- Application Programming Interfaces of the C5 microkernel. These APIs are fully compatible with Jaluna-1.

Linux Components:

- Linux Driver and Module Programming Interfaces.
- Linux process-level APIs.

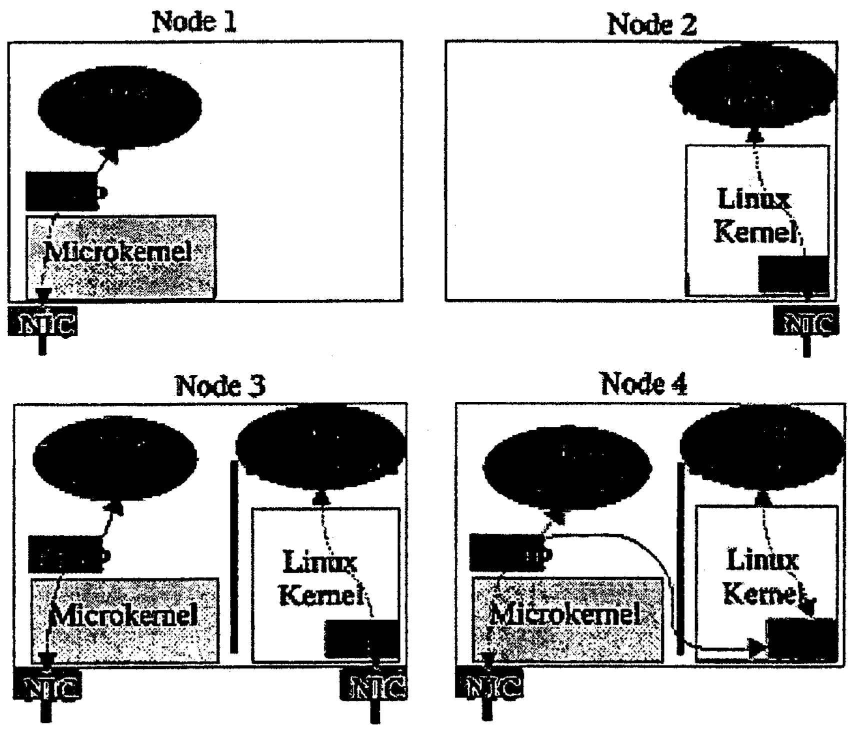
1.4. Deployment Configuration

Jaluna-2 PR1 architecture is highly flexible and can be adapted to various customer configurations. The Figure 1-3 illustrates with four nodes the possible configurations of Jaluna-2 PR1.

Jaluna-2 PR1 may be configured with a single kernel. This kernel might be the C5 microkernel, or the Linux kernel as deployed on Node 2. Or both kernels can be configured on the same node as done on Nodes 3 and 4.

When both kernel are configured and two network interfaces (labeled NIC in the figure) are available (node 3), each side can have its interface. When one network interface only is available (node 4), it is controlled by the C5 side and the Real-Time UDP/IP stack is configured to redirect the traffic from C5 to Linux.

Figure 1-3. Jaluna-2 PR1 Deployment



Each side from node 1, 3 and 4 can communicate with each other using the ISC (not represented). In the case of a single Linux kernel (node 2), the ISC is not available.

1.5. Jaluna-2 PR1 Supported Hosts and Targets

1.5.1. Hosts

- Linux Red Hat 7.3

1.5.2. Targets Families and Boards

Jaluna-2 PR1 supports two target processor families. Other families might be added on customer requests basis. For each family, Jaluna-2 PR1 comes with the implementation of at least one reference target board and provides a complete set of well-defined interfaces allowing customers to port Jaluna-2 PR1 to other boards in the same target family.

The Boot Kernel Interface (BKI) of Jaluna-2 PR1 allows to customize the boot method.

1.5.2.1. PowerPC Family

The supported processor targets are:

- Motorola PowerPC 750
- Motorola PowerPC 74x0
- Motorola PowerPC 604

The supported BSP's are:

- mcp7xx for MCP750, MCPN750 and MCPN765
- genesis2 for MVME3600

1.5.2.2. Intel Family

As processor, Jaluna-2 PR1 supports the IA-32 Intel Architecture. On recent boards where the BIOS setup has a PIC/APIC mode selection, the PIC mode has to be chosen.

The supported BSP is the PC Reference BSP.

Chapter 2. Cross-Development Environment

2.1. C and C++ Compilers

The Cross-Development Environment (CDE) includes several cross-compilers. The architecture of gcc leads to have one compiler per host, target OS, and target CPU family. In Jaluna-2 PR1, the host is linux-x86, the target OS is either C5 or Linux, and the CPU family is either x86 or powerpc. This leads to four cross-compilers being provided. To ensure correct operation of the Linux system software, gcc version 2.95.3 has been retained for Linux. On the C5 side, gcc version 3.2 is provided, the same as in Jaluna-1. The same compilers are used for building kernels and applications, both for C5 and Linux.

Include files and libraries are provided in a transparent mode: the cross-compilation tools (gcc, ld, ...) are able to find and use the development software (includes, libraries) which has been installed on the host without the need to explicitly indicate its location (no need of -I*includes* or -L*libraries* parameters).

2.2. C and C++ Symbolic Debugger

The debugger is GNU gdb 5.2.1 with the Insight graphical user interface.

gdb provides source-level host target symbolic debugging of the C5 microkernel, of actors and drivers running on the C5 microkernel, of the Linux system and of Linux processes.

On the C5 side, gdb connects to the Debug Server, a host program which is aware of the C5 microkernel and actors. The Debug Server connects to the target through a serial line.

On the Linux side, to debug the kernel gdb connects with the kgdb module through a serial line. To debug applications, either gdb connects with a gdb server running on the target, or gdb runs directly on the target to perform a standard local debugging. For the latter case, gdb version 5.1 is provided.

2.3. Tools and Utilities

The Jaluna-2 PR1 cross-development environment is made of tools coming from the open-source world and tools developed by Jaluna.

2.3.1. Cross-Development System

The CDS component contains the cross-development tools coming from the open-source world. It consists of the cross-compilers, the RPM tool and GNU tools. These tools are:

- gcc 3.2 and 2.95.3
- binutils 2.13
- rpm 4.0.4

2.3.2. dev_tools Component

The dev_tools component contains the cross-development tools developed by Jaluna (for example: used to configure the system image) as well as the environment used to control the execution of the CDS component tools (for example: configuration files, compilation rules).

2.3.3. Configuration Tools

Configuration tools are applied at three stages of the entire build process of the system:

- Configuration of the build of the system in terms of components.
- Configuration of the C5 microkernel and of the Linux kernel in terms of features, drivers, modules and tunables.
- Configuration of the Jaluna-2 PR1 Root File System in terms of packages.

Configuring the system as a collection of components is the first operation that must be performed. It can be done with the **configure** tool. The main available components are the C5 microkernel, the Linux kernel, and the Linux root file system. Components can be used either in source or in binary form. Dependencies or possible incompatibilities between components are checked.

Configuration of the C5 microkernel is based on the **configurator** tool. It allows to specify microkernel features, tunables and drivers that will be included in the system image.

Configuring the Linux kernel can be done with the standard Linux configuration tools (**make xconfig, make menuconfig, make config**). It allows to indicate the features/modules that will be installed in the Linux kernel.

Configuring the Jaluna-2 PR1 Root File System consists in specifying the packages that will be installed in the target root file system. A configuration file `linux-root.conf`, located at the root of the build directory, lists all packages and groups of packages. Dependencies are managed by RPM.

Chapter 3. C5 Microkernel Component

3.1. Configurability

The C5 microkernel is modular from the ground up. Modules are either fundamental such as scheduling or memory management, or optional, such as logging. Microkernel configuration is an essential step to accurately meet the needs of a given application or environment. A set of constraints and dependencies help ensure the consistency of configurations. Two ready-to-use configuration profiles are provided. Modules come as binaries so that the configuration does not require a recompilation.

3.2. Inter-System Communication

The Inter-System Communication is based on the C5 bus communication framework, and more specifically on the following drivers:

`syscom`

Low level communication driver over local memory.

`buscom`

Low level communication driver over the CompactPCI bus.

`busmux`

Communication driver over syscom or buscom. It implements the ISC API.

`buseth`

Ethernet emulation over the ISC.

Thc API for using the ISC is either the busmux device driver interface, or the Real-Time UDP/IP API over the Ethernet emulation.

3.3. Actor Loading

C5 actors may be embedded in a Jaluna-2 PR1 system image and start up automatically when C5 boots up, or they may be launched using the Linux-side **muxloader** utility. This program loads the binaries from disk, and connects with a server running on the C5 side which loads the actors in C5 memory and starts them up. **muxloader** is also able to list and stop running actors, and to start up embedded actors that were configured not to start up automatically.

3.4. Inter-Process Communications

The C5 IPC addresses communication between C5 actors, either locally on the same node, or remotely over an Ethernet link. It offers three reliable, location-transparent and message-oriented services:

- A synchronous Remote Procedure Call service, with which all messages and replies are guaranteed to be delivered exactly once.
- An asynchronous communication service in which applications exchange messages through end-points named "ports"
- A broadcasting service in which applications send messages to groups of ports.

Multiple parallel remote communications between two sites are achieved through the same inter-site connection on both sites. Hence in addition to being reliable, these services are scalable and highly efficient.

3.5. List of Features

The C5 microkernel features supported by Jaluna-2 PR1 are listed in the following table:

Table 3-1. C5 Features

Component	Name
Virtual address spaces	VIRTUAL_ADDRESS_SPACE
Microkernel statistics	MKSTAT
Linux Monitor	LINUX_MON
Semaphores	SEM
Event flag sets	EVENT
Mutexes	MUTEX
Mutexes with priority inversion avoidance	RTMUTEX
Periodic timers	TIMER
Thread and actor virtual timer	VTIMER

Component	Name
Date and time of day (Intel family only)	DATE
Real-time clock	RTC
Location-transparent C5 IPCs	IPC
Remote C5 IPCs	IPC_REMOTE
Mailbox-based communications	MIPC
Private per-thread data	PRIVATE_DATA
Console message logging	LOG
Profiling and benchmark support	PERF
System Monitoring	MON
Debug Agent for remote system debugging	SYSTEM_DEBUG
Real-Time UDP/IP stack	UDP
ISC over bus	BUSCOM
ISC over local memory	SYSCOM
AltiVec SIMD support	ALTIVEC

3.6. Libraries

Table 3-2. C5 Libraries

Description	Library
Small footprint C library	libebd
C library	libc
C++ library	libstdc++
Encryption/decryption library	libcrypt
Blackbox static library	libblackbox
Mathematical library	libm

Chapter 4. Linux Kernel

4.1. Configurability

Linux configuration within Jaluna-2 PR1 is identical to regular Linux configuration. However, Jaluna-2 PR1 specific modules may be configured in order to take advantage of the microkernel services from the Linux world.

Jaluna-2 PR1 specific Linux kernel modules offer the following services:

- the Inter-System Communication
- the Linux Monitoring pat(see Section 1.2.6)
- the Remote Block Device (see Section 1.2.6)

4.2. Inter System Communication

The bus communication stack is implemented in several Linux kernel modules. This stack has been ported from C5 code using a implementation of the C5 Device-Kernel Interface over Linux kernel primitives, on which the bus communication code runs practically unchanged. This ensure complete interoperability between both stacks.

The API for using the ISC from the Linux kernel is the busmux device driver interface. For Linux processes, the ISC exports an interface through the /proc file system.

4.3. Features

As Jaluna-2 PR1 includes a standard Linux kernel, all Linux kernel features are available and work as they do in standard Linux. Jaluna-2 PR1 provides a default kernel configuration for each of the supported families, that is, Intel and PowerPC.

Chapter 5. Linux Root File System

5.1. Configurability

Jaluna-2 PR1 uses a flexible cross package management to adapt the root file system to the exact needs in term of size and functionality.

The Jaluna-2 PR1 root file system can be profiled from a minimal (about 20 MB) to a full featured system for native development (about 130 MB). This can be configured with three potential levels of granularity:

Profile

The configuration profiles are global to the whole system (C5 microkernel, Linux kernel and root file system configuration).

Group

Packages are organized in groups.

Package

For a critical usage it is possible to populate the root file system on a per package basis.

All the configuration can be done manually using a rpm utility (adapted for host-target operation) or automatically by selecting packages in a configuration file.

5.2. Features

The Jaluna-2 PR1 root file system is targeted for an embedded use, this implies some differences with a standard Linux root file system to deal with diskless requirements.

The Jaluna-2 PR1 root file system includes the following features:

- The root file system can be generated, configured and installed from the host, without target access, and does not require to have root privileges.
- All components (utilities, libraries, ...) of the root file system are provided as a set of RPM packages.
- All packages of the root file system are also provided in source format (SRPM) and can be easily recompiled from the host.

- Source packages are mostly issued from the Red Hat 7.3 Linux distribution and adapted to allow cross-compilation and cross-installation.
- The initialization process of the Jaluna-2 PR1 system on the target has been modified from standard Linux to fit embedded needs (diskless initialization, no hardware probing for fast reboot, ...).
- The root file system can be mounted and shared through NFS or from read-only devices (Flash or cdrom devices, ...) using devfs facilities.

5.3. Packages

Jaluna-2 PR1 uses the RPM package management to add, retrieve or remove components in the root file system.

The Jaluna-2 PR1 root file system is built based on the following source RPM packages, organized into the following groups:

BASIC

  The basic Linux utilities needed for a minimal use.

NET

  All network utilities and daemons which are not included in BASIC.

DEV

  Development utilities, libraries and headers.

HA

  High availability utilities and daemons.

DOCS

  Documentation

Each source RPM package produces one or more binary RPM packages, that are also organized in BASIC, NET, DEV, HA and DOCS groups.